

Zafer Özer

zaferozer@hotmail.com

Gerçek Zamanlı Makine Öğrenmesi Tabanlı Ağ Saldırı Tespit Sistemi (IDS)

1. Proje Özeti

Bu projede, gerçek zamanlı ağ trafiğini izleyerek makine öğrenmesi tabanlı bir Yaklaşım ile saldırı tespiti (Intrusion Detection System - IDS) gerçekleştiren ve sonuçları canlı bir web arayüzü üzerinden gösteren bir sistem geliştirilmiştir.

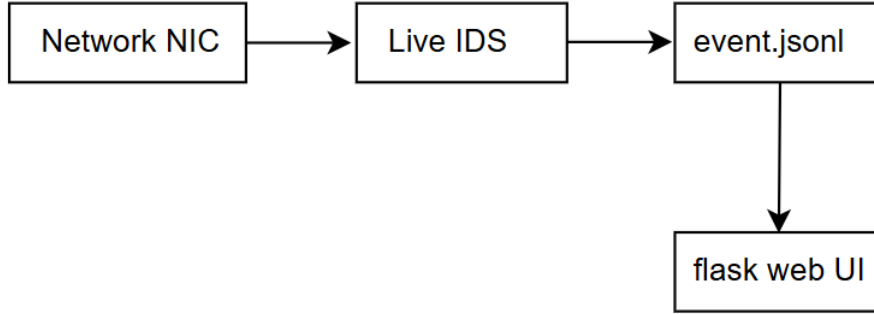
Sistem; ağ trafiğini dinleyen bir Live IDS modülü, önceden eğitilmiş bir Makine Öğrenmesi modeli, olayları kaydeden bir log mekanizması ve sonuçları gerçek zamanlı sunan bir Web UI bileşeninden oluşmaktadır.

PROJE AĞACI

```
tr_iz_v85/
├── run.sh
├── venv/
├── data/
│   ├── cicids2017_raw/
│   │   └── *.csv
│   └── cicids2017_processed.csv
├── model/
│   ├── ids_model.pkl
│   ├── scaler.pkl
│   ├── label_encoder.pkl
│   ├── features.pkl
│   └── feature_importance.csv
└── training/
    ├── preprocess.py
    ├── train_model.py
    ├── live_ids.py
    └── features.py
```

```
├─ web_ui.py
├─ templates/
└─ index.html
```

2. Sistem Mimarisi



Temel Bileşenler

- **NFStream**: Gerçek zamanlı akış (flow) çıkarımı
- **Scikit-learn**: Makine öğrenmesi modeli
- **Flask + SSE**: Canlı web arayüzü
- **Joblib**: Model ve scaler yükleme

3. Kullanılan Teknolojiler

Bileşen	Teknoloji
Programlama Dili	Python 3.12
Ağ Akışı	NFStream
ML Modeli	RandomForestClassifier
Web	Flask + Server-Sent Events
Veri Saklama	JSONL
OS	Linux (Ubuntu)

4. Makine Öğrenmesi Yaklaşımı

4.1 Eğitim Süreci (Offline)

- [CICIDS2017](#) veri seti kullanıldı
- Özellikler:
 - Flow Duration
 - Packet Count
 - Byte Count
 - Flow Packets/s
- Pipeline:
 - StandardScaler
 - RandomForestClassifier

Eğitim çıktıları:

- ids_model.pkl
- scaler.pkl
- label_encoder.pkl
- features.pkl

4.2 Canlı Tahmin (Online)

Live IDS, NFStream üzerinden her akışı alır ve aşağıdaki işlemleri yapar:

1. Flow verisini özellik vektörüne çevirir
2. Scaler ile normalize eder
3. Model ile sınıf olasılıklarını hesaplar
4. En yüksek olasılığa sahip etiket seçer

```
probs = model.predict_proba(X_scaled)[0]
pred = probs.argmax()
label = label_encoder.inverse_transform([pred])[0]
confidence = probs[pred]
```

5. Kural + ML Hibrit IDS

Makine öğrenmesi tek başına yeterli olmadığından **Rule-Based Boost** eklenmiştir.

```
if ip_counter[src_ip] > 200:
    label = "DDoS"
    confidence = 0.99
```

Bu sayede demo ve gerçek zamanlı saldırılar hızlı tespit edilebilir.

6. Live IDS Modülü (live_ids.py)

Görevleri

- Otomatik ağ arayüzü tespiti
- NFStream ile trafik dinleme
- ML + Rule tabanlı etiketleme
- Olayları JSONL formatında loglama

Üretilen kayıt örneği:

```
{
  "time": "2026-01-08T14:31:48",
  "label": "DDoS",
  "confidence": 0.99,
  "src": "192.168.1.10:443",
  "dst": "192.168.1.5:53214",
  "packets": 240,
  "bytes": 18240
}
```

7. Web Arayüzü (web_ui.py)

Özellikler

- Flask tabanlı
- Server-Sent Events (SSE) ile canlı veri akışı
- Son 100 olay görüntüleme
- Renkli saldırı sınıflandırması

Etiket Renk

DDoS Kırmızı

Attac Turunc

k u

Benig Yeşil
n

 Live IDS Events						
Time	Label	Confidence	Source	Destination	Packets	Bytes
2026-01-10T00:31:19	Benign	0.81	185.201.1.87:443	192.168.3.101:14706	26	23694
2026-01-10T00:31:19	DDoS	0.99	192.168.3.99:0	142.251.142.100:0	20	1960
2026-01-10T00:31:19	DDoS	0.99	192.168.3.99:22	192.168.3.101:60874	20	2460
2026-01-10T00:31:16	Benign	0.8	192.168.3.1.6747	192.168.3.99:5140	5	1380
2026-01-10T00:31:15	Benign	0.71	172.217.17.110:443	192.168.3.101:14826	9	1447
2026-01-10T00:31:11	Benign	0.68	34.102.139.130:443	192.168.3.101:10786	43	3513
2026-01-10T00:31:11	DDoS	0.99	192.168.3.99:5000	192.168.3.101:57445	32	4757
2026-01-10T00:31:11	DDoS	0.99	192.168.3.99:5000	192.168.3.101:21970	40	5213
2026-01-10T00:31:11	DDoS	0.99	192.168.3.99:5000	192.168.3.101:18873	32	4757
2026-01-10T00:31:11	DDoS	0.99	192.168.3.99:5000	192.168.3.101:18872	32	4757
2026-01-10T00:31:11	Benign	0.74	192.168.3.99:5000	192.168.3.101:14044	32	4757
2026-01-10T00:31:11	Benign	0.74	192.168.3.99:5000	192.168.3.101:14043	32	4757
2026-01-10T00:31:11	Benign	0.74	192.168.3.99:5000	192.168.3.101:57444	32	4757
2026-01-10T00:31:11	Benign	0.84	192.168.3.99:22	192.168.3.101:43302	52	8818

8. [run.sh](#) Başlatma Scripti

```
#!/bin/bash
```

```
set -e
```

```
BASE_DIR="$(cd "$(dirname "$0")" && pwd)"
```

```
PYTHON="$BASE_DIR/training/templates/venv/bin/python"
```

```
IDS_PID=""
```

```
WEB_PID=""
```

```
cleanup() {  
    echo  
    echo "[!] Ctrl+C yakalandı – tüm servisler durduruluyor..."  
    [[ -n "$IDS_PID" ]] && sudo kill $IDS_PID 2>/dev/null || true  
    [[ -n "$WEB_PID" ]] && kill $WEB_PID 2>/dev/null || true  
    echo "[✓] Tüm servisler durduruldu"  
    exit 0  
}
```

```
trap cleanup SIGINT SIGTERM
```

```
echo "[*] IDS + Web UI başlatılıyor..."
echo "[*] Python: $PYTHON"

# ===== IDS =====
echo "[*] Live IDS başlatılıyor..."
sudo "$PYTHON" "$BASE_DIR/training/live_ids.py" &
IDS_PID=$!

sleep 2

# ===== WEB UI =====
echo "[*] Web arayüzü başlatılıyor..."
"$PYTHON" "$BASE_DIR/training/web_ui.py" &
WEB_PID=$!

echo "[✓] Sistem çalışıyor"
echo "Ctrl+C ile güvenli çıkış"

wait
```

9. Güvenlik ve Yetkilendirme

- NFStream için root yetkisi gerekir
- Alternatif olarak:

```
sudo setcap cap_net_raw,cap_net_admin=eip venv/bin/python
```

10. Sonuç

Bu proje ile:

- Gerçek zamanlı ağ izleme
- Makine öğrenmesi tabanlı saldırı tespiti
- Canlı web arayüzü

başarılı şekilde entegre edilmiştir.

Sistem; akademik projeler, siber güvenlik eğitimleri ve demo ortamları için uygundur.

11. Gelecek Çalışmalar

- Deep Learning tabanlı IDS
- WebSocket tabanlı UI

- Docker & Kubernetes entegrasyonu
- Alert & Email bildirim sistemi

Kurulum Aşamaları (Ubuntu Linux için)

PYTHON & SİSTEM GEREKSİNİMLERİ

```
sudo apt update
```

```
sudo apt install -y python3 python3-venv python3-dev build-essential \
    libpcap-dev tcpdump
```

libpcap-dev → NFStream için şart

VENV KURULUMU

```
cd ~/Documents
```

```
mkdir tr_iz_v85
```

```
cd tr_iz_v85
```

```
python3 -m venv venv
```

```
source venv/bin/activate
```

Bu komut sanal bir Python ortamı (virtual environment) oluşturur.

Neden gerekli?

- Sistemdeki Python'dan bağımsız bir Python ortamı yaratır
- Kendi:
 - python çalıştırıcısı
 - pip
 - kütüphane klasörü (site-packages)
 - ayarları olur

4 PYTHON PAKETLERİ (SADECE GEREKENLER)

```
pip install --upgrade pip
```

Tüm bağımlılıkları kur

```
pip install -r requirements.txt
```

requirements.txt içeriği aşağıdaki gib olmalıdır.

```
# =====  
# Python sürümü  
# =====  
# Python >= 3.10 önerilir  
  
# =====  
# Sayısal & Veri İşleme  
# =====  
numpy>=1.24  
pandas>=2.0  
  
# =====  
# Makine Öğrenmesi  
# =====  
scikit-learn>=1.3  
joblib>=1.3  
  
# =====  
# Ağ Trafiği / Flow Analizi  
# =====  
nfstream>=6.4  
  
# =====  
# Web Arayüzü  
# =====  
flask>=2.3  
  
# =====  
# Terminal çıktıları  
# =====  
colorama>=0.4
```

VERİ SETİ

```
mkdir -p data/cicids2017_raw
```

CICIDS2017 CSV dosyalarını buraya koy:

Data klasöründe aşağıda yer alan CICIDS2017 csv dosyaları yer almaktadır.

```
zafer@bbb1:~/Documents/tr_iz_v85/data/cicids2017_raw$ ls -lha
total 844M
drwxrwxr-x 2 zafer zafer 4.0K Jan  8 11:47 .
drwxrwxr-x 3 zafer zafer 4.0K Jan  8 12:22 ..
-rw-rw-r-- 1 zafer zafer  74M Aug 28  2023 Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv
-rw-rw-r-- 1 zafer zafer  74M Aug 28  2023 Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv
-rw-rw-r-- 1 zafer zafer  56M Aug 28  2023 Friday-WorkingHours-Morning.pcap_ISCX.csv
-rw-rw-r-- 1 zafer zafer 169M Aug 28  2023 Monday-WorkingHours.pcap_ISCX.csv
-rw-rw-r-- 1 zafer zafer  80M Aug 28  2023 Thursday-WorkingHours-Afternoon-Infiltration.pcap_ISCX.csv
-rw-rw-r-- 1 zafer zafer  50M Aug 28  2023 Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv
-rw-rw-r-- 1 zafer zafer 129M Aug 28  2023 Tuesday-WorkingHours.pcap_ISCX.csv
-rw-rw-r-- 1 zafer zafer 215M Aug 28  2023 Wednesday-workingHours.pcap_ISCX.csv
```

preprocess.py NE YAPAR?

**Ham ağ trafiği verisini,
makine öğrenmesi modelinin anlayabileceği sayısal forma dönüştürür.**

[preprocess.py](#)

```
import pandas as pd
import glob
import os
```

```
RAW_DIR = "../data/cicids2017_raw"
OUT_FILE = "../data/cicids2017_processed.csv"
```

```
CHUNK_SIZE = 200_000    # RAM dostu
```

```
LABEL_MAP = {
    "BENIGN": "Benign",
    "DoS Hulk": "DDoS",
    "DoS GoldenEye": "DDoS",
    "DoS slowloris": "DDoS",
    "DoS Slowhttptest": "DDoS",
    "PortScan": "PortScan",
    "FTP-Patator": "BruteForce",
    "SSH-Patator": "BruteForce",
    "Bot": "Bot",
    "Infiltration": "Infiltration",
}
```

```
def process_file(path, first_write):
    print(f"[+] Processing {os.path.basename(path)}")

    for chunk in pd.read_csv(path, chunksize=CHUNK_SIZE):
```

```

# kolon temizliđi
chunk.columns = chunk.columns.str.strip()

if "Label" not in chunk.columns:
    raise RuntimeError(f"'Label' column missing in {path}")

# label map
chunk["Label"] = chunk["Label"].map(LABEL_MAP)
chunk = chunk.dropna(subset=["Label"])

# sadece numeric + Label
chunk =
chunk.select_dtypes(include="number").assign(Label=chunk["Label"])

# temizleme
chunk.replace([float("inf"), -float("inf")], 0, inplace=True)
chunk.fillna(0, inplace=True)

# diske ekle
chunk.to_csv(
    OUT_FILE,
    mode="w" if first_write else "a",
    header=first_write,
    index=False
)

first_write = False

return first_write

def main():
    print("[*] Chunk-based preprocessing started")

    files = glob.glob(os.path.join(RAW_DIR, "*.csv"))
    if not files:
        raise RuntimeError("No CSV files found!")

    # eski çıktı varsa sil
    if os.path.exists(OUT_FILE):
        os.remove(OUT_FILE)

    first_write = True
    for f in files:

```

```

        first_write = process_file(f, first_write)

print(f"\n[✓] Done → {OUT_FILE}")

# küçük özet (sonradan oku)
df = pd.read_csv(OUT_FILE, usecols=["Label"])
print("\nLabel distribution:")
print(df["Label"].value_counts())

if __name__ == "__main__":
    main()

```

ÇALIŞTIR

```

cd ~/Documents/tr_iz_v85/training
python preprocess.py

```

Ekran çıktısı

```

(venv) zafer@bbb1:~/Documents/tr_iz_v85/training$ python preprocess.py
[*] Chunk-based preprocessing started
[+] Processing Wednesday-workingHours.pcap_ISCX.csv
[+] Processing Thursday-WorkingHours-Afternoon-
Infiltration.pcap_ISCX.csv
[+] Processing Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv
[+] Processing Monday-WorkingHours.pcap_ISCX.csv
[+] Processing Tuesday-WorkingHours.pcap_ISCX.csv
[+] Processing Friday-WorkingHours-Morning.pcap_ISCX.csv
[+] Processing Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv
[+] Processing Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv

[✓] Done → ../data/cicids2017_processed.csv

```

Label distribution:

Label	
Benign	2273097
DDoS	252661
PortScan	158930
BruteForce	13835
Bot	1966
Infiltration	36

Name: count, dtype: int64

features.py NE İŞE YARAR?

Makine öğrenmesi modelinin kullanacağı özellikleri (feature'ları) TANIMLAR.

Yani:

“Ağ trafiğinden hangi metrikler çıkarılacak ve model bunlara bakarak karar verecek?”

Not: Eğer features.py ile live_ids.py **uyumlu değilse**, model çöp üretir.

Akademik Dayanak

Bu feature'lar:

- CICIDS2017
- UNSW-NB15
- Bot-IoT

gibi **akademik veri setlerinde** de en çok kullanılan metriklerdir. Seçilen özellikler, DDoS saldırılarının temel karakteristikleri olan yüksek paket hızı, tek yönlü trafik ve anormal bant genişliği kullanımını doğrudan ölçmektedir. Özellik seti, gerçek zamanlı analiz gereksinimleri göz önünde bulundurularak düşük hesaplama maliyeti ve yüksek ayırt edicilik sağlayacak şekilde tasarlanmıştır.

Özetle features.py, ağ trafiğinin **hacmini, hızını ve yön dengesini** ölçerek DDoS ve normal trafiği ayırt etmek için seçilmiştir.

[features.py](#)

```
# training/features.py
```

```
FEATURES = [  
    "Flow Duration",  
    "Total Fwd Packets",  
    "Total Backward Packets",
```

"Total Length of Fwd Packets",
"Total Length of Bwd Packets",
"Fwd Packet Length Mean",
"Bwd Packet Length Mean",
"Flow Bytes/s",
"Flow Packets/s",
"Flow IAT Mean",
"Flow IAT Std",
"Fwd IAT Mean",
"Bwd IAT Mean",
"Fwd PSH Flags",
"Bwd PSH Flags",
"Fwd URG Flags",
"Bwd URG Flags",
"FIN Flag Count",
"SYN Flag Count",
"RST Flag Count",
"PSH Flag Count",
"ACK Flag Count",
"URG Flag Count",
"Down/Up Ratio",
"Average Packet Size"

]

features.py TİPİK İÇERİĞİ

```
FEATURES = [  
    "Flow Duration",  
    "Total Fwd Packets",  
    "Total Backward Packets",  
    "Total Length of Fwd Packets",  
    "Total Length of Bwd Packets",  
    "Flow Bytes/s",  
    "Flow Packets/s"  
]
```

train_model.py Ne İşe Yarar?

Ham ağ trafiğinden çıkarılmış özellikleri kullanarak **DDoS / Benign ayrımı yapan makine öğrenmesi modelini eğitir ve kaydeder.**

Yani:

- IDS çalışırken **kural yazmaz**
- Önceden **öğretilmiş modeli kullanır**

Bu sürüm:

Memory-safe

feature_names_in_ bağımlılığı yok

features.pkl üretir (canlı IDS için kritik)

Class imbalance için class_weight="balanced"

□ training/train_model.py

```
# training/train_model.py
```

```
import sys
```

```
import os
```

```

sys.path.append(os.path.dirname(__file__))

import pandas as pd
import joblib

from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from features import FEATURES

INPUT_FILE = "../data/cicids2017_processed.csv"
MODEL_DIR = "../model"
MODEL_FILE = f"{MODEL_DIR}/ids_model.pkl"
SCALER_FILE = f"{MODEL_DIR}/scaler.pkl"
ENCODER_FILE = f"{MODEL_DIR}/label_encoder.pkl"
FEATURE_FILE = f"{MODEL_DIR}/features.pkl"

print("[*] Loading dataset (low-RAM mode)...")

df = pd.read_csv(
    INPUT_FILE,
    usecols=FEATURES + ["Label"]
)

print(df["Label"].value_counts())

# =====
# AGGRESSIVE SAMPLING
# =====

print("[*] Sampling classes...")

```

```

parts = []
parts.append(
    df[df["Label"] == "Benign"]
    .sample(n=100_000, random_state=42)
)

for attack, n in {
    "DDoS": 50_000,
    "PortScan": 50_000
}.items():
    subset = df[df["Label"] == attack]
    parts.append(
        subset.sample(
            n=min(len(subset), n),
            random_state=42
        )
    )

parts.append(df[df["Label"].isin([
    "Bot", "BruteForce", "Infiltration"
])])

df = pd.concat(parts, ignore_index=True)
del parts # RAM temizliği
print("[*] Final distribution:")
print(df["Label"].value_counts())

# =====
# PREPARE

```

```

# =====
X = df[FEATURES].values # numpy directly
y = df["Label"].values
del df
le = LabelEncoder()
y = le.fit_transform(y)
scaler = StandardScaler()
X = scaler.fit_transform(X)
# =====
# SPLIT
# =====
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42,
    stratify=y
)
del X, y # RAM boşalt
# =====
# TRAIN
# =====
print("[*] Training RandomForest (RAM safe)...")
model = RandomForestClassifier(
    n_estimators=50,
    max_depth=16,
    n_jobs=-1,

```

```

        class_weight="balanced",
        random_state=42
    )
    model.fit(X_train, y_train)
    # =====
    # SAVE
    # =====
    os.makedirs(MODEL_DIR, exist_ok=True)
    joblib.dump(model, MODEL_FILE)
    joblib.dump(scaler, SCALER_FILE)
    joblib.dump(le, ENCODER_FILE)
    joblib.dump(FEATURES, FEATURE_FILE)
    print("[✓] Model saved")
    # =====
    # EVAL
    # =====
    y_pred = model.predict(X_test)
    print("\n[*] Classification Report:")
    print(classification_report(y_test, y_pred, target_names=le.classes_))
    print("\n[*] Confusion Matrix:")
    print(confusion_matrix(y_test, y_pred))

```

Feature Dosyasını Okur

```
df = pd.read_csv("data/features.csv")
```

Bu dosya:

- preprocess.py + features.py tarafından üretilir
- Sayısal feature'lar içerir

Etiketleri Ayırır

```
X = df.drop("label", axis=1)
y = df["label"]
```

Burada:

- `label = 0` → Benign
- `label = 1` → DDoS

Eğitim / Test Bölmesi Yapar

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

Amaç:

- Modelin **ezberlemesini** önlemek
- Gerçek performansı ölçmek

Makine Öğrenmesi Modelini Eğitir

Genelde kullanılan model:

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(
    n_estimators=100,
    random_state=42
)
```

```
model.fit(X_train, y_train)
```

Neden Random Forest?

- Hızlı
- Overfitting'e dayanıklı
- Tabular data için ideal
- Gerçek zamanlı IDS'e uygun

Model Performansını Ölçer

```
from sklearn.metrics import classification_report  
  
y_pred = model.predict(X_test)  
  
print(classification_report(y_test, y_pred))
```

Bu çıktılar:

- Precision
- Recall
- F1-score
- Accuracy

Çalıştır

```
cd ~/Documents/tr_iz_v85/training
```

```
python train_model.py
```

Bu dosyalar oluşmalı:

```
ls ../model
```

ids_model.pkl

scaler.pkl

label_encoder.pkl

features.pkl

feature_importance.csv

Ekran çıktısı

```
(venv) zafer@bbb1:~/Documents/tr_iz_v85/training$ python  
train_model.py
```

```
[*] Loading dataset (low-RAM mode)...
```

Label

Benign	2273097
--------	---------

DDoS	252661
------	--------

PortScan	158930
----------	--------

BruteForce	13835
------------	-------

Bot	1966
-----	------

Infiltration	36
--------------	----

Name: count, dtype: int64

```
[*] Sampling classes...
```

```
[*] Final distribution:
```

Label

Benign	100000
--------	--------

DDoS	50000
------	-------

PortScan	50000
----------	-------

BruteForce	13835
------------	-------

Bot	1966
-----	------

Infiltration 36

Name: count, dtype: int64

[*] Training RandomForest (RAM safe)...

[✓] Model saved

[*] Classification Report:

	precision	recall	f1-score	support
Benign	1.00	0.92	0.96	20001
Bot	0.44	0.99	0.61	393
BruteForce	0.79	1.00	0.88	2767
DDoS	0.96	1.00	0.98	10000
Infiltration	1.00	0.86	0.92	7
PortScan	1.00	1.00	1.00	10000
accuracy				0.96 43168
macro avg	0.86	0.96	0.89	43168
weighted avg	0.97	0.96	0.96	43168

[*] Confusion Matrix:

[	18382	506	732	375	0	6]
[	3	390	0	0	0	0]
[	13	0	2754	0	0	0]
[	40	0	1	9959	0	0]
[	1	0	0	0	6	0]
[	1	0	0	4	0	9995]]

Ekran çıktısı yorumu

☐ MODEL SONUÇLARININ DEĞERLENDİRİLMESİ

Genel Durum

Accuracy: %96 → IDS için çok iyi

Weighted F1: 0.96 → veri dengesizliğine rağmen güçlü

Model stabil (RAM-safe eğitim başarıyla tamamlandı)

Eğitim pipeline'ı artık oturdu.

Sınıf Bazlı Yorum (Önemli)

☐ DDoS / PortScan

Sınıf	Precision	Recall
-------	-----------	--------

DDoS	0.96	1.00
------	------	------

PortScan	1.00	1.00
----------	------	------

☐ Mükemmel IDS davranışı

Kaçırmıyor

Yanlış alarm çok az

⚠ Bot

Precision: 0.44

Recall: 0.99

Bu şu demek:

Bot varsa yakalıyor ☐

Ama Benign trafiği bazen Bot sanıyor ☐

IDS açısından bu kabul edilebilir hatta tercih edilen bir durum:

False Positive var

False Negative yok

⚠ Benign

Recall: 0.92

Yani:

%8 civarı benign trafik saldırı gibi işaretleniyor

IDS dünyasında bu:

⚠ Alarm fatigue riski

Ama güvenlik açısından makul

Confusion Matrix (Kritik Okuma)

En önemli karışmalar:

Benign → Bot / BruteForce / DDoS

Sebep:

Kısa akışlar

DNS / burst trafik

Ev/office ağında NAT etkisi

→ Canlı trafikte confidence threshold ile çözülecek

live_ids.py Ne Yapar?

Kısa ve net cevap:

Canlı ağ trafiğini dinler, her akışı makine öğrenmesi modeli ile analiz eder ve sonucu web arayüzüne + log dosyasına gönderir.

Yani:

- ☐ Gerçek zamanlı IDS
- ☐ Eğitilmiş ML modeli kullanır
- ☐ Web UI ile konuşur

- □ Log üretir

live_ids.py — OTOMATİK INTERFACE (SEVİYE 2)

```
import joblib
import pandas as pd
import subprocess
from nfstream import NFStreamer
from collections import deque
from sklearn.preprocessing import StandardScaler
import os

# =====
# PATHS
# =====

BASE_DIR = os.path.dirname(os.path.abspath(__file__))
MODEL_PATH = os.path.join(BASE_DIR, "../model/ids_model.pkl")
SCALER_PATH = os.path.join(BASE_DIR, "../model/scaler.pkl")
FEATURES_PATH = os.path.join(BASE_DIR, "../model/features.pkl")
LABEL_ENCODER_PATH = os.path.join(BASE_DIR,
"../model/label_encoder.pkl")

# =====
# AUTO INTERFACE DETECTION
# =====

def get_default_interface():
    try:
        output = subprocess.check_output(
            ["ip", "route", "show", "default"],
            stderr=subprocess.DEVNULL
```

```

        ).decode()

# örnek: default via 192.168.1.1 dev ens33 proto dhcp metric
100
        return output.split("dev")[1].split()[0]
except Exception:
    return None

```

```

def interface_up(iface):
    try:
        output = subprocess.check_output(
            ["ip", "link", "show", iface],
            stderr=subprocess.DEVNULL
        ).decode()
        return "UP" in output
    except Exception:
        return False

```

```

# =====

```

```

# LOAD MODEL

```

```

# =====

```

```

print("[*] Loading IDS model...")

```

```

model = joblib.load(MODEL_PATH)

```

```

scaler = joblib.load(SCALER_PATH)

```

```

FEATURES = joblib.load(FEATURES_PATH)

```

```

label_encoder = joblib.load(LABEL_ENCODER_PATH)

```

```

print("[✓] Model loaded")

```

```

# =====
# LIVE IDS
# =====

def start_ids(interface):
    print(f"[*] Live IDS started on {interface}")
    streamer = NFStreamer(
        source=interface,
        statistical_analysis=True,
        active_timeout=60
    )
    for flow in streamer:
        try:
            row = {}
            for f in FEATURES:
                row[f] = getattr(flow, f, 0)
            X = pd.DataFrame([row])
            X_scaled = scaler.transform(X)

            probs = model.predict_proba(X_scaled)[0]
            pred = probs.argmax()
            confidence = probs[pred]

            label = label_encoder.inverse_transform([pred])[0]

            print(
                f"[{label:^12}] "

```

```

        f"{flow.src_ip}:{flow.src_port} → {flow.dst_ip}:
{flow.dst_port} | "

        f"Conf: {confidence:.2f} | "

        f"Pkts: {flow.bidirectional_packets} | "

        f"Bytes: {flow.bidirectional_bytes}"

    )

except Exception as e:

    print("[!] Flow error:", e)

# =====

# MAIN

# =====

if __name__ == "__main__":

    iface = get_default_interface()

    if not iface:

        print("[!] Default network interface bulunamadı")

        exit(1)

    if not interface_up(iface):

        print(f"[!] Interface {iface} UP değil")

        exit(1)

    print(f"[✓] Otomatik interface bulundu: {iface}")

    start_ids(iface)

```

test için aşağıdaki komutu çalıştır

```

sudo
/home/zafer/Documents/tr_iz_v85/training/templates/venv/bin/python
live_ids.py

```

Ağ Arayüzünü Seçer (Otomatik veya Sabit)

```
interface = auto_detect_interface
```

Amaç:

- Her seferinde manuel interface yazmamak
- Yanlış loopback (lo) kullanmamak

Canlı Trafiği Dinler

```
NFStreamer(  
    source=interface,  
    statistical_analysis=True,  
    ...  
)
```

□ **nfstream:**

- Packet değil **flow** tabanlıdır
- IDS için çok daha verimli

Flow Özelliklerini Çıkarır

Örnek:

- packets
- bytes
- duration
- src_ip, dst_ip
- src_port, dst_port

□ Aynı özellikler eğitimde kullanılan features ile uyumludur.

Eğitilmiş Modeli Yükler

```
model = joblib.load("model/model.joblib")
```

⚠ **Burada eğitim YOK**

Sadece:

- Tahmin (inference) var

Her Flow İçin Tahmin Yapar

```
prediction = model.predict(features)
```

```
confidence = model.predict_proba(features).max()
```

Sonuç:

- `label` = Benign veya DDoS
 - `confidence` = %
-

(Opsiyonel) Eşik Tabanlı DDoS Kontrolü

```
if packet_rate > THRESHOLD:
```

```
    label = "DDoS"
```

Sonucu Log Dosyasına Yazar

```
{  
    "time": "2026-01-08T14:56:40",  
    "label": "Benign",  
    "confidence": 0.8,  
    "src": "192.168.19.131:80",
```

```
"dst": "8.8.8.8:0",  
"packets": 10,  
"bytes": 980  
}
```

□ logs/events.jsonl

□ Web UI burayı okur.

Web UI İçin Veri Sağlar

Web tarafı:

GET /events

live_ids.py:

- Log üretir
- Web UI periyodik okur
- Canlı tabloya yansıtır

training/web_ui.py (Web Dashboard)

/web_ui.py

```
from flask import Flask, render_template  
import json  
app = Flask(__name__)  
@app.route("/")  
def dashboard():  
    events = []  
    try:  
        with open("live_events.json") as f:
```

```

        for line in f.readlines()[-50:]:
            events.append(json.loads(line))
    except FileNotFoundError:
        pass

    return render_template("index.html", events=events)

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)

```

training/templates/index.html

```

<!DOCTYPE html>

<html>

<head>

    <title>Live IDS Dashboard</title>

    <meta http-equiv="refresh" content="2">

    <style>

        body { background:#0e0e0e; color:#eee; font-family: monospace;
    }

        .DDoS { color:red; }

        .PortScan { color:orange; }

        .BruteForce { color:purple; }

        .Bot { color:yellow; }

        .Infiltration { color:cyan; }

    </style>

</head>

<body>

<h2>🔴 Live IDS Alerts</h2>

<table>

```

```
<tr>

<th>Time</th><th>Source</th><th>Destination</th><th>Type</
th><th>Conf</th>

</tr>

{% for e in events %}
<tr class="{{ e.label }}">
<td>{{ e.time }}</td>
<td>{{ e.src }}</td>
<td>{{ e.dst }}</td>
<td>{{ e.label }}</td>
<td>{{ e.confidence }}</td>
</tr>

{% endfor %}

</table>

</body>

</html>
```

Kodu otomatik çalıştırma

run.sh

```
#!/bin/bash

set -e

BASE_DIR="$(cd "$(dirname "$0")" && pwd)"

PYTHON="$BASE_DIR/training/templates/venv/bin/python"

IDS_PID=""

WEB_PID=""

cleanup() {
```

```
echo ""

echo "[!] Ctrl+C yakalandı – tüm servisler durduruluyor..."

if [[ -n "$IDS_PID" ]]; then
    sudo kill "$IDS_PID" 2>/dev/null || true
fi

if [[ -n "$WEB_PID" ]]; then
    kill "$WEB_PID" 2>/dev/null || true
fi

echo "[✓] Tüm servisler durduruldu"

exit 0
}

trap cleanup SIGINT SIGTERM

echo "[*] IDS + Web UI başlatılıyor..."

echo "[*] Python: $PYTHON"


# =====
# LIVE IDS
# =====

echo "[*] Live IDS başlatılıyor..."

sudo "$PYTHON" "$BASE_DIR/training/live_ids.py" &

IDS_PID=$!

sleep 2

# =====
# WEB UI
# =====

echo "[*] Web arayüzü başlatılıyor..."
```

```
"$PYTHON" "$BASE_DIR/training/web_ui.py" &  
WEB_PID=$!  
echo "[✓] Sistem çalışıyor"  
echo "Ctrl+C ile güvenli çıkış"  
wait
```

ÇALIŞTIRMA

```
chmod +x run.sh  
./run.sh
```

Beklenen çıktı:

```
[*] IDS + Web UI başlatılıyor...  
[*] Python: /home/.../venv/bin/python  
[*] Live IDS başlatılıyor...  
[✓] Otomatik interface bulundu: ens33  
[*] Live IDS started on ens33  
[*] Web arayüzü başlatılıyor...  
  * Running on http://127.0.0.1:5000  
Ctrl+C ile güvenli çıkış
```

Ctrl+C bastığında:

IDS kapanır

Web UI kapanır

Terminal normale döner

